

INSTITUTE FOR COMPUTER APPLICATIONS IN  
SCIENCE AND ENGINEERING (ICASE)

7N-64-CR

210977

348

A DESCENT METHOD FOR THE  
UNIFORM SOLUTION TO OVER-DETERMINED  
SYSTEMS OF LINEAR EQUATIONS

A. K. Cline

(NASA-CR-179996) A DESCENT METHOD FOR THE  
UNIFORM SOLUTION TO OVER-DETERMINED SYSTEMS  
OF LINEAR EQUATIONS (ICASE) 24 P

N89-70670

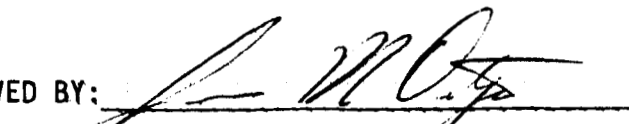
00/64    Unclass    0210977

August 23, 1974

A DESCENT METHOD FOR THE  
UNIFORM SOLUTION TO OVER-DETERMINED  
SYSTEMS OF LINEAR EQUATIONS

A. K. Cline

APPROVED BY:

  
James M. Ortega, Director  
Institute for Computer Applications  
in Science and Engineering

This work was begun at the National Center for Atmospheric Research, which is sponsored by the National Science Foundation, and completed under NASA Grant 47-102-001 while the author was in residence at ICASE.

# ABSTRACT

Given a system of linear equations with more equations than unknowns, we seek to determine that vector of unknowns which minimizes the norm of the residual of the system in the uniform sense. A method is presented which obtains this solution after a finite number of trial solutions have been examined in a sequence in which the residual norm decreases with each successive step. The implementation of the method exploits efficient matrix decomposition updating schemes resulting in reduced computation times when compared with a presently popular method.

## 1. INTRODUCTION

Suppose we are given an  $m \times n$  real matrix  $A$  (with  $m > n$ ), a real  $m$ -vector  $b$  and we seek to determine a real  $n$ -vector  $x^*$  such that

$$||Ax^* - b|| \leq ||Ax - b||$$

for all real  $n$ -vectors  $x$  (where  $||\cdot||$  indicates the uniform norm). Such an  $x^*$  is termed a "uniform" ("Chebychef", "Minimax", " $L_\infty$ ") solution to the overdetermined system denoted by " $Ax \approx b$ ".

That such a solution  $x^*$  exists is easy to show. We first notice that  $||y - b||$  is a continuous function of  $y = Ax$ . The set  $\{y | y = Ax, ||y - b|| \leq ||b||\}$  is a compact subset of the (at most)  $n$  dimensional subspace  $\{Ax, x \in R^n\}$  of  $R^m$ , hence contains a minimum  $y^*$  of  $||y - b||$  and this minimum is obviously a minimum over the entire subspace. By selecting an  $x^*$  such that  $Ax^* = y^*$ , we have  $||Ax^* - b|| \leq ||Ax - b||$  for all  $x \in R^n$ .

If furthermore, it is assumed that every  $n \times n$  submatrix of  $A$  is non-singular then there exists a unique solution  $x^*$ . This assumption on  $A$  is called the Haar condition; should it not hold,  $x^*$  may still be unique for a fixed  $b$ . However there always exists some  $b$  for which the solution is not unique. Henceforth it will be assumed that  $A$  satisfies the Haar condition.

A proof that the Haar condition does guarantee a unique solution may be found in various approximation theory texts (e.g., Cheney [7]). The algorithm presented here provides a constructive proof. This method begins with any initial  $x$  and proceeds to determine a finite sequence of  $x$ 's reducing the quantity  $||Ax - b||$  at each step, until  $x^*$  is reached where  $||Ax - b||$  can no longer be reduced.

Section 2 includes some initial definitions and discussion as well as a

geometrical interpretation of the descent algorithm. The algorithm is analytically presented in Section 3. Section 4 discusses the actual efficient implementation of the algorithm. Computational experience and comparison with another technique is mentioned in Section 5. In Section 6, the modifications to the algorithm allowing the addition of linear equality and inequality constraints are briefly discussed.

## 2. THE RESIDUAL AND THE GEOMETRY OF THE ALGORITHM

For any given  $n$ -vector  $x$ , the residual  $Ax - b$  will be denoted by  $r(x)$  (and occasionally simply by  $r$ ). The quantity  $||r(x)||$  we seek to minimize is the maximum of the magnitudes of the  $m$  components of  $r(x)$ . We may partition the set of all residuals,  $R = \{r(x) | x \in R^n\}$ , into two distinct classes according to the number of components of the vector which are maximal in magnitude:

$$R_1 = \{r(x) | n \text{ or less of the components of } r(x) \text{ are equal to } ||r(x)|| \text{ in magnitude}\}$$

$$R_2 = \{r(x) | n + 1 \text{ or more of the components of } r(x) \text{ are equal to } ||r(x)|| \text{ in magnitude}\}$$

If we select a particular  $x \in R^n$  and assume all such selections are equally likely, then with probability one the corresponding residual  $r(x)$  lies in  $R_1$ . In the descent algorithm, if we begin with an initial trial solution whose residual lies in  $R_1$ , and thus has  $k$  ( $\leq n$ ) components of maximal magnitude, we shall move to a second trial solution with  $k + 1$  or more residual components of maximal magnitude. Subsequently, at each step the number of such components of the residual must increase by at least one, with the result that after at most  $n$  steps an  $x$  has been determined whose residual belongs to  $R_2$ . All trial solutions subsequent to this have residuals in  $R_2$ , although the number of maximal magnitude components does not necessarily increase from step to step.

At each step the quantity  $||r(x)||$  decreases; hence the name "descent method". The iteration continues until a point is reached at which no descent is possible, implying that the solution has been reached.

Graphically we may consider the subset of  $R^n \times R$  defined by  $\{(x, \theta) | \theta \geq ||r(x)||, x \in R^n\}$ . This set, being the intersection of the  $2m$  half-spaces

$$\{(x, \theta) | \theta \geq a_i x - b_i; x \in R^n\} \quad i = 1, \dots, m$$

and

$$\{(x, \theta) | \theta \geq -(a_i x - b_i); x \in R^n\} \quad i = 1, \dots, m$$

(where  $a_i$  denotes the  $i^{\text{th}}$  row of  $A$  and  $b_i$  the  $i^{\text{th}}$  element of  $b$ ), is a convex polytope. Points on its surface must have the form  $(x, ||r(x)||)$  and it is clear that if we introduce the notion of "down" corresponding to decrease in the last component, that the lowest point on the polytope is precisely  $(x^*, ||r(x^*)||)$ . We may think graphically that the solution to the problem is equivalent to determination of the minimal point of the polytope. The descent algorithm to be presented produces a finite sequence of points on the surface of the polytope which descend to this minimal point.

Using the nomenclature of Berge [6, page 169], we define a "privileged line" of a polytope  $P$  through a point  $a$ , as a line whose intersection with  $P$  contains  $a$  on the interior. If  $V_a$  denotes the linear manifold formed by the privileged lines of  $P$  through  $a$ , then the intersection of  $V_a$  and  $P$  is a "face" and the "order" of the face is the dimension of  $V_a$ . Faces of order zero are vertices (thus vertices are exactly the extreme points of the polytope) and clearly every point on the surface of the given polytope is contained in a face of order at most  $n$ . There is one face of order  $n + 1$ : the interior of the polytope. Since all faces of lower order are intersections of a finite number of half-spaces the total number of faces is finite. This fact provides the basis for the proof of finite termination of the algorithm: each descent ends at the minimal point of a face, hence only finitely many descents are possible.

### 3. THE DESCENT ALGORITHM

For a given vector  $x$ , the residual  $r(x)$  must have some  $k$  of its components of maximal magnitude (where  $1 \leq k \leq m$ ) and  $m - k$  strictly submaximal. Let  $M(x)$  be the set of those  $k$  indices corresponding to maximal components. Further, let  $M^+(x)$  denote the same set of indices except given the sign of the corresponding residual component: i.e., if  $i \in M(x)$ , then  $i \in M^+(x)$  if  $r_i(x) > 0$  and  $-i \in M^+(x)$  if  $r_i(x) < 0$ .

The following lemma and theorem provide a correspondence between polytope faces and the sets  $M^+(x)$ .

LEMMA 1: A line  $L = \{(x + \lambda \Delta x, ||r(x)|| + \lambda z) | \lambda \in \mathbb{R}\}$  is a privileged line of the polytope through  $(x, ||r(x)||)$  if and only if

$$a_i \Delta x = \text{sgn } r_i(x) \cdot z$$

for all  $i \in M(x)$ .

PROOF: For  $L$  to be a privileged line through  $(x, ||r(x)||)$  we must have

$$|a_i(x + \lambda \Delta x) - b_i| \leq ||r(x)|| + \lambda z$$

for  $i = 1, \dots, m$  and all  $\lambda$  in some neighborhood of 0. Thus for the same values of  $i$  and  $\lambda$  we have

$$\text{sgn } r_i(x) \cdot (a_i x + \lambda a_i \Delta x - b_i) \leq ||r(x)|| + \lambda z.$$

For  $i \in M(x)$  we have

$$\text{sgn } r_i(x) \cdot (a_i x - b_i) = ||r(x)||,$$



and thus it follows that

$$\operatorname{sgn} r_i(x) \cdot \lambda a_i \Delta x \leq \lambda z.$$

This holds for positive and negative values of  $\lambda$ , resulting in

$$\operatorname{sgn} r_i(x) \cdot a_i \Delta x = z$$

for  $i \in M(x)$ .

Now suppose  $a_i \Delta x = \operatorname{sgn} r_i(x) \cdot z$  holds for all  $i \in M(x)$ . Then

$$\operatorname{sgn} r_i(x) \cdot (a_i x + \lambda a_i \Delta x - b_i) = ||r(x)|| + \lambda z;$$

thus

$$|a_i(x + \lambda \Delta x) - b_i| \leq ||r(x)|| + \lambda z$$

for  $\lambda$  in a neighborhood of 0. For  $i \notin M(x)$

$$|r_i(x)| < ||r(x)||;$$

thus

$$|a_i(x + \lambda \Delta x) - b_i| \leq |r_i(x)| + |\lambda| \cdot |a_i \Delta x| \leq ||r(x)|| + \lambda z$$

for  $\lambda$  in some neighborhood of 0. Combining these we have

$$|a_i(x + \lambda \Delta x) - b_i| \leq ||r(x)|| + \lambda z$$

for  $i = 1, \dots, m$  and  $\lambda$  in some neighborhood of 0 which is equivalent to  $L$  being a privileged line of the polytope. ■

THEOREM 1: The two faces generated by the privileged lines through  $(x_1, ||r(x_1)||)$  and  $(x_2, ||r(x_2)||)$  respectively, are equal if and only if  $M^+(x_1) = M^+(x_2)$ .

PROOF: We may assume  $x_1 \neq x_2$ . Suppose first that the two faces are the same. The line passing through  $(x_1, ||r(x_1)||)$  and  $(x_2, ||r(x_2)||)$  must then be a privileged line at both points. From the lemma

$$\text{sgn } r_i(x_1) \cdot a_i \Delta x = ||r(x_1)|| - ||r(x_2)||$$

for  $i \in M(x_1)$  and

$$\text{sgn } r_i(x_2) \cdot a_i \Delta x = ||r(x_1)|| - ||r(x_2)||$$

for  $i \in M(x_2)$ . Suppose  $M^+(x_1) \neq M^+(x_2)$ . Thus, without loss of generality we may assume there is an  $i^* \in M^+(x_1) \sim M^+(x_2)$ , and either  $|r_{|i^*|}(x_2)| < ||r(x_2)||$  or  $\text{sgn } r_{|i^*|}(x_2) \neq \text{sgn } r_{|i^*|}(x_1)$ .

We have

$$\begin{aligned} ||r(x_1)|| - ||r(x_2)|| &= \text{sgn } r_{|i^*|}(x_1) a_{|i^*|} x_1 - \text{sgn } r_{|i^*|}(x_1) a_{|i^*|} x_2 \\ &= \text{sgn } r_{|i^*|}(x_1) \cdot (a_{|i^*|} x_1 - b_{|i^*|}) - \text{sgn } r_{|i^*|}(x_1) (a_{|i^*|} x_2 - b_{|i^*|}) \\ &= ||r(x_1)|| - \text{sgn } r_{|i^*|}(x_1) r_{|i^*|}(x_2). \end{aligned}$$

Thus

$$||r(x_2)|| = \text{sgn } r_{|i^*|}(x_1) \cdot r_{|i^*|}(x_2)$$

which is a contradiction whether  $|r_{|i^*|}(x_2)| < ||r(x_2)||$  or  $\text{sgn } r_{|i^*|}(x_2) \neq \text{sgn } r_{|i^*|}(x_1)$ .

Now suppose that  $M^+(x_1) = M^+(x_2)$ . From the lemma, we may see that the line

through  $(x_1, ||r(x_1)||)$  and  $(x_2, ||r(x_2)||)$  is a privileged line at both points. The faces generated by the privileged lines at each of the points must include the opposite point and thus the faces must coincide. ■

Using Theorem 1, we see that we may bound the number of faces by bounding the number of possible sets  $M^\pm$ . This bound is clearly  $\sum_{k=1}^m \binom{m}{k} 2^k = 3^m - 1$ ; in practice however we find the number of faces to be on the order of  $m$ .

### 3. THE DESCENT ALGORITHM

For a given vector  $x$ , we shall define a direction of descent from  $x$  as any  $n$ -vector  $\Delta x$  for which  $||r(x + \lambda \Delta x)|| < ||r(x)||$  for all sufficiently small, but positive, values of  $\lambda$ . The following theorem is the basis for determination of directions of descent.

THEOREM 2: If the system of inequalities

$$\text{sgn}(r_i(x)) \cdot a_i \Delta x < 0$$

for all  $i \in M(x)$ , has a solution, then  $\Delta x$  is also a direction of descent from  $x$ . If no solution exists then no descent is possible and  $x = x^*$ , the minimizer of  $||r(x)||$ .

PROOF: Suppose first that  $x \neq x^*$ . We must prove that a solution to the system of linear inequalities

$$\text{sgn}(r_i(x)) a_i \Delta x < 0$$

for  $i \in M(x)$ , exists.

Since  $||r(x^*)|| < ||r(x)||$ , we have for  $i \in M(x)$ ,

$$\begin{aligned} \text{sgn}(r_i(x))(a_i x^* - b_i) &\leq ||r(x^*)|| \\ &< ||r(x)|| = \text{sgn}(r_i(x))(a_i x - b). \end{aligned}$$

Subtracting we obtain

$$\text{sgn}(r_i(x)) a_i (x^* - x) < 0;$$

thus,  $x^* - x$  always satisfies the inequalities. Alternatively suppose there

exists a solution  $\Delta x$  so that for  $i \in M(x)$ ,

$$\text{sgn}(r_i(x)) a_i \Delta x < 0.$$

For sufficiently small, positive  $\lambda$ , we have

$$|r_i(x + \lambda \Delta x)| = |r_i(x)| + \lambda \text{sgn}(r_i(x)) a_i \Delta x < |r_i(x)| = ||r(x)||$$

for  $i \in M(x)$ . For  $i \notin M(x)$   $|r_i(x)| < ||r(x)||$ . Thus for sufficiently small positive  $\lambda$  (in fact for

$$\lambda < \frac{||r(x)|| - |r_i(x)|}{|a_i \Delta x|},$$

$|r_i(x + \lambda \Delta x)| < ||r(x)||$ . Combining the results for  $i \in M(x)$  we can say for  $\lambda$  sufficiently small, positive and independent of  $i$

$$|r_i(x + \lambda \Delta x)| < ||r(x)||;$$

thus

$$||r(x + \lambda \Delta x)|| < ||r(x)||;$$

and  $\Delta x$  is a direction of descent. ■

If we have determined a direction of descent  $\Delta x$  from  $x$ , we are guaranteed some positive  $\lambda$  so that the step from  $x$  to  $x + \lambda \Delta x$  results in a decrease in the residual norm. Since  $\Delta x$  must satisfy the inequalities of Theorem 2, we may examine the proof to see that  $\lambda$  satisfying

$$\lambda = \min \left\{ \min_{i \in M(x)} \left( \frac{-r_i(x)}{a_i \Delta x} \right), \min_{\substack{i \notin M(x) \\ a_i \Delta x \neq 0}} \left( \frac{||r(x)|| - |r_i(x)|}{|a_i \Delta x|} \right) \right\}$$

suffices. We could consider an algorithm where at each trial solution  $x$ , the direction of descent  $\Delta x$  was determined as some solution to the linear inequalities,

then using the  $\lambda$  selected from the above,  $x$  is replaced by  $x + \lambda \Delta x$ , and the process repeats. With this algorithm we may guarantee that  $||r(x)||$  decreases at each step but there is no guarantee of finite convergence (i.e., determination of the solution in a finite number of steps) or even any convergence at all. It is possible for the sequence of solutions from this algorithm to have a set of accumulation points, none of which is  $x^*$ .

To produce an algorithm with finite convergence, we must be more careful about the selection of  $\Delta x$  and  $\lambda$ .

The following theorem provides such a careful selection.

THEOREM 3: Suppose for all  $i \in M(x)$

$$\text{sgn}(r_i(x)) a_i \Delta x \leq -||r(x)||.$$

Define  $M'(x) = \{i \in M(x) | \text{sgn}(r_i(x)) a_i \Delta x = -||r(x)||\}$ , and for  $i \notin M'(x)$ ,

$$\lambda_i = \frac{||r(x)|| - \text{sgn}(a_i \Delta x + r_i(x)) r_i(x)}{||r(x)|| + \text{sgn}(a_i \Delta x + r_i(x)) a_i \Delta x},$$

$$\hat{\lambda} = \min_{i \in I'} \lambda_i; \text{ and } \hat{x} = x + \hat{\lambda} \Delta x.$$

Then it follows that

- i)  $0 < \hat{\lambda} \leq 1$
- ii)  $||r(\hat{x})|| = (1 - \hat{\lambda}) ||r(x)||$
- iii)  $M'(x) \subsetneq M(\hat{x})$

PROOF: For i) it suffices to show that  $0 < \lambda_i \leq 1$  for all  $i \notin M'(x)$ . Notice first that since  $|r_i(x)| \leq ||r(x)||$ ,  $||r(x)|| - \text{sgn}(a_i \Delta x + r_i(x)) r_i(x) \geq 0$ . Furthermore if equality were to hold we would have

$$||r(x)|| = |r_i(x)| = \text{sgn}(a_i \Delta x + r_i(x)) r_i(x)$$

which implies that  $\text{sgn}(a_i \Delta x + r_i(x)) = \text{sgn}(r_i(x))$  and that  $i \in M(x)$ . By virtue of  $i \notin M'(x)$ , however,

$$\text{sgn}(r_i(x)) a_i \Delta x < - ||r(x)|| = -\text{sgn}(r_i(x))(r_i(x))$$

i.e.,  $\text{sgn}(r_i(x))(a_i \Delta x + r_i(x)) < 0$  and

$$\begin{aligned} & \text{sgn}(a_i \Delta x + r_i(x))(a_i \Delta x + r_i(x)) \\ &= |a_i \Delta x + r_i(x)| < 0 \end{aligned}$$

which is a contradiction. Henceforth we may assume that the numerator in the expression for  $\lambda_i$  is strictly positive. That the denominator is also strictly positive follows because for it to be non-positive we would have

$$-||r(x)|| > \text{sgn}(a_i \Delta x + r_i(x)) a_i \Delta x.$$

Since  $||r(x)|| > 0$ ,  $\text{sgn}(a_i \Delta x + r_i(x))$  must then be  $-\text{sgn}(a_i \Delta x)$  which is equivalent to  $|a_i \Delta x| < |r_i(x)|$ ; thus  $|a_i \Delta x| < ||r(x)||$  but this contradicts the above inequality.

Having now shown that  $\lambda_i > 0$  we show  $\lambda_i \leq 1$ . If it were the case that  $\lambda_i > 1$ , we would have

$$\begin{aligned} & ||r(x)|| - \text{sgn}(a_i \Delta x + r_i(x)) r_i(x) \\ & > ||r(x)|| + \text{sgn}(a_i \Delta x + r_i(x)) \lambda_i \Delta x \end{aligned}$$

$$\text{i.e., } 0 > \text{sgn}(a_i \Delta x + r_i(x))(a_i \Delta x + r_i(x)) > |a_i \Delta x + r_i(x)|$$

which is a contradiction.

Turning to conclusion ii) we shall show that for all  $i$

$$|r_i(\hat{x})| \leq (1 - \hat{\lambda}) ||r(x)||$$

and for  $i \in M'(x)$

$$|r_i(\hat{x})| = (1 - \hat{\lambda}) ||r(x)||$$

For the first we need recall that  $|r_i(x)| \leq ||r(x)||$  and thus the curve  $|r_i(x + \lambda \Delta x)|$  for  $\lambda \geq 0$  is dominated initially by  $(1 - \lambda) ||r(x)||$ . For  $i \notin M'(x)$ , the first, positive intersection of these curves is exactly at  $\lambda = \lambda_i$ , thus since  $\hat{\lambda} \leq \lambda_i$ ,  $|r_i(x + \lambda \Delta x)| \leq (1 - \hat{\lambda}) ||r(x)||$ . For  $i \in M'(x)$ ,  $r_i(x + \lambda \Delta x) = r_i(x) + \lambda a_i \Delta x = (1 - \lambda) r_i(x)$ . Thus  $|r_i(x + \lambda \Delta x)| = (1 - \lambda) |r_i(x)| = (1 - \lambda) ||r(x)||$ . For all  $0 < \lambda \leq 1$ , and in particular for  $\lambda = \hat{\lambda}$  we have

$$|r_i(\hat{x})| = (1 - \hat{\lambda}) ||r(x)||.$$

For the third part of the conclusion we need only show that for some  $i \notin M'(x)$ ,  $|r_i(\hat{x})| = ||r(\hat{x})||$  (since it has already been shown that  $M'(x) \subset M(\hat{x})$ ). But for some  $M'(x)$ ,  $\lambda_i = \hat{\lambda}$  and as was stated previously  $|r_i(x + \lambda_i \Delta x)| = (1 - \lambda_i) ||r(x)||$ ; thus  $|r_i(\hat{x})| = ||r(\hat{x})||$ . ■

The descent algorithm can now be described using the construction of the theorem. Given any point  $x$ , the system of inequalities

$$\text{sgn}(r_i(x)) a_i \Delta x \leq - ||r(x)|| \quad i \in M(x)$$

is solved in such a way that the subset  $M'(x)$  of indices for which equality holds, is as large as possible. Notice that if  $M(x)$  has  $n$  or less indices that it is possible to have equality for all indices of  $M(x)$  (i.e.,  $M'(x) = M(x)$ ) and from conclusion (iii) of Theorem 3 it follows that  $M(\hat{x})$  has at least one more component than  $M(x)$ . We see that in  $n$  or less such steps we



determine an  $x$  such that  $M(x)$  has  $n+1$  or more components. This is the initial phase of the descent. In the second phase more care must be shown in the selection of the direction of descent.

We shall show that a vertex of the polytope associated with the problem can be reached in a finite number of steps, (usually  $n$ ) and then from one vertex, another is found also in a finite number of steps (usually one). Using the fact that the number of vertices is finite, we may conclude that the "lowest" vertex on the polytope, that one corresponding to the solution, is located in a finite number of steps.

Lemma 2: If  $(x, ||r(x)||)$  is not a vertex on the polytope then we may solve

$$\text{sgn}(v_i(x)) a_i \Delta x = -||r(x)||$$

for all  $i \in M(x)$  and determine  $\hat{x}$ , as in Theorem 3, such that  $M(x) \neq M(\hat{x})$ . Thus (since  $M(x)$  may contain at most  $m$  elements) eventually a vertex must be reached by recursively descending in this manner.

PROOF: Since  $(x, ||r(x)||)$  is not a vertex there is at least one privileged line passing through it. But according to Lemma 1, this implies a solution to  $a_i \Delta x = \text{sgn } v_i(x) z$  for all  $i \in M(x)$  for some  $\Delta x$  and  $z$ , which is equivalent to  $\text{sgn}(v_i(x)) a_i \Delta x = -||v(x)||$  for suitably rescaled  $\Delta x$  and  $z$ . ■

At a vertex we must have at least  $n+1$  indices in  $M(x)$ . We shall determine a  $\Delta x$  as before with the set  $M'(x)$  containing at least  $n$  elements. How this is done in the case of more than  $n+1$  indices in  $M(x)$  will not be discussed here, since this case occurs exceedingly rarely in practice and the techniques are simple generalizations. The case of exactly  $n+1$  will be discussed in detail.

Let  $k$  be any element of  $M(x)$  and  $\Delta x$  the unique solution to

$$\text{sgn}(v_i(x))a_i \Delta x = -||v(x)||$$

for  $i \in M(x) \sim k$ . If  $\text{sgn}(v_k(x)) a_k \Delta x \leq -||r(x)||$  then the descent step of Theorem 3 can be performed without further computation. If  $\text{sgn}(v_k(x)) a_k \Delta x > -||r(x)||$  then let  $y$  be the  $n$ -vector that expresses  $\text{sgn}(r_k(x)) a_k$  as a linear combination of the vectors  $\text{sgn}(r_i(x))a_i$ ,  $i \in M(x) \sim k$ . That  $y$  exists and has no zero component follows from the Haar condition. Since

$$\text{sgn}(r_k(x))a_k = \sum_{i \in M(x) \sim k} y_i \text{sgn}(r_i(x))a_i$$

(where we have assumed an indexing of  $y$  corresponding to the indices of  $M(x) \sim k$ ) if we were to select some  $j \in M(x) \sim k$  it would follow that

$$\begin{aligned} \text{sgn}(r_j(x))a_j &= \sum_{i \in M(x) \sim j \sim k} (-y_i/y_j) \text{sgn}(r_i(x))a_i \\ &\quad + 1/y_j \text{sgn}(r_k(x))a_k. \end{aligned}$$

Thus if now we solve the system

$$\text{sgn}(r_i(x))a_i \overline{\Delta x} = -||v(x)||$$

for  $i \in M(x) \sim j$ , we would have  $\text{sgn}(r_j(x))a_j \overline{\Delta x}$

$$\begin{aligned} &= \sum_{i \in M(x) \sim j \sim k} (-y_i/y_j) \text{sgn}(r_i(x))a_i \overline{\Delta x} \\ &\quad + 1/y_j \text{sgn}(r_k(x))a_k \overline{\Delta x} \\ &= \sum_{i \in M(x) \sim j \sim k} (-y_i/y_j)(-||r(x)||) + (1/y_j)(-||r(x)||) \\ &= (1 - S + y_j)/y_j (-||r(x)||) \end{aligned}$$

where  $S = \sum_{i \in M(x) \sim k} y_i$ .

We would like to determine  $j$  so that

$$\text{sgn}(r_j(x)) a_j \Delta x \leq -||r(x)||$$

which is equivalent to

$$(1 - S + y_j)/y_j \geq 1,$$

i.e.,  $(1 - S)/y_j \geq 0.$

We then select  $j$  so that the quantity  $(1 - S)/y_j$  is maximized. However, with a little inspection we can see that  $1 - S > 0$ , so it suffices to maximize  $1/y_j$  (i.e., determine minimum  $y_j$  for  $y_j > 0$ ). If  $1 - S \leq 0$ , thus  $S \geq 1$ , notice that

$$\begin{aligned} \text{sgn}(r_k(x)) a_k \Delta x &= \sum_{i \in M(x)} y_i \text{sgn}(r_i(x)) a_i \Delta x \\ &= S (-||r(x)||) \leq -||r(x)||, \end{aligned}$$

contradicting our assumption that

$$\text{sgn}(r_k(x)) a_k \Delta x > -||r(x)||.$$

Returning to the determination of  $j$ , if all  $y_j < 0$ , then  $x$  is the solution to the problem since descent is impossible. If there would be such a direction of descent  $v$  then it would satisfy

$$0 > \text{sgn}(r_i(x)) a_i v \text{ for } i \in M(x).$$

But then

$$0 > \text{sgn}(r_k(x))a_k v = \sum_{i \in M(x) \sim k} y_i \text{sgn}(r_i(x))a_i v > 0,$$

which is a contradiction.

As a result we have shown that either a direction of descent  $\Delta x$  or  $\overline{\Delta x}$  may be determined or that the solution has been reached. This is summarized in the following theorem.

Theorem 4: If  $(x, ||r(x)||)$  is a vertex of the polytope such that the set  $M(x)$  has exactly  $n+1$  components and  $k$  is an element of  $M(x)$  then one of the following holds

i) Letting  $\Delta x$  satisfy

$$\text{sgn}(r_i(x))a_i \Delta x = -||r(x)|| \quad i \in M(x) \sim k$$

results in

$$\text{sgn}(r_k(x))a_k \Delta x \leq -||r(x)||$$

so that  $\Delta x$  yields a direction of descent as in Theorem 3 with  $M(x) \sim k \subset M'(x)$ .

ii) Letting  $\Delta x$  satisfy

$$\text{sgn}(r_i(x))a_i \Delta x = -||r(x)|| \quad i \in M(x) \sim k$$

results in

$$\text{sgn}(r_K(x))a_K \Delta x > -||r(x)||$$

so that  $\Delta x$  is not a direction of descent. However letting  $y$  satisfy

$$\text{sgn}(r_K(x))a_K = \sum_{i \in M(x) \sim k} y_i \text{sgn}(r_i(x))a_i$$

and selecting  $j$  so that  $y_j > 0$  but  $y_j \leq y_i$  for all  $y_i > 0$ , then by solving

$$\text{sgn}(r_i(x))a_i \overline{\Delta x} = -||r(x)|| \quad i \in M(x) \sim j$$

results in  $\text{sgn}(r_j(x))a_j \overline{\Delta x} \leq -||r(x)||$  so that  $\overline{\Delta x}$  yields a direction of descent as in Theorem 3 with  $M(x) \sim j \subset M'(x)$ . If however no  $y_j > 0$  then  $x = x^*$  the solution.

#### 4. COMPUTATIONAL ASPECTS OF THE ALGORITHM

The primary computation required in the descent algorithm is that of solving a sequence of square linear systems. If we denote such a system by  $By = d$ , then it may also be required to solve a system of the form  $B^T w = f$  (which is the case when vertices are reached). Furthermore another system  $B'y' = d'$ , may need to be solved at the subsequent step,  $B'$  differing from  $B$  in only one row. We seek to provide numerically stable methods for solving these problems.

An obvious approach is to perform an LU factorization of  $B$  at each step and make no use of the relation between  $B$  and  $B'$ . Thus we have  $B = LUP$  where  $L$  is lower triangular,  $U$  is unit upper triangular, and  $P$  is a suitable permutation matrix. This factorization is the result of Gaussian elimination by columns with partial pivoting. To solve  $By = d$  we need only solve the lower triangular system  $\bar{L}\bar{y} = d$ , the upper triangular system  $U\bar{y} = \bar{y}$ , and then let  $y = P^T\bar{y}$ . To solve the transposed problem  $B^T w = f$  we may use the same factorization: Let  $\bar{w} = Pf$ , solve the lower triangular system  $U^T\bar{w} = \bar{w}$ , and finally solve the upper triangular system  $L^T w = \bar{w}$ .

We see that the LU factorization is useful, but it requires approximately  $n^3/3$  multiplications (and the same number of additions) to determine. An obvious method for reducing the computation is to notice that the first  $k-1$  steps of the factorization are essentially independent of rows  $k$  through  $n$  (and in particular row  $k$ ). If we perform  $k-1$  steps of the factorization, we have the form  $BP_k^T = L_k U_k$  where  $P_k$  is a permutation matrix and  $L_k$  has the structure

$L^1$	0
$L^2$	$L^3$
$L^4$	$L^5$

(where  $L^1$  is  $k-1 \times k-1$  lower triangular and  $\ell^2$  is  $k-1 \times 1$  and  $\ell^3$  is  $n-k+1 \times 1$  and  $U_k$  has the structure

$U^1$	$U^2$
0	I

(where  $U^1$  is  $k-1 \times k-1$  upper triangular). First notice that if  $B'P_k'^T = L_k'^T U_k'$  represents the first  $k-1$  steps of factorization of  $B'$ , then  $P_k = P_k'$  since interchanging of columns in the first  $k-1$  steps is based only on elements of the first  $k-1$  rows, and  $B$  and  $B'$  are identical in these elements. Notice, second, that

$$c_1' = (c_1' U^{1-1}) U^1 + (c_2' - c_1' U^{1-1} U^2) 0$$

$$c_2' = (c_1' U^{2-1}) U^2 + (c_2' - c_1' U^{2-1} U^2) I$$

(Where  $c_1'$  represents the first  $k-1$  components of  $cP_k'^T$  and  $c_2'$  the remainder), or in matrix form

$$cP_k'^T = [c_1' U^{1-1}, c_2' - c_1' U^{1-1} U^2] U_k.$$

Thus, letting  $\ell^{2'} = c_1' U^{1-1}$  and  $\ell^{3'} = c_2' - c_1' U^{1-1} U^2$  we have

$$B'P_k'^T = \begin{array}{|c|c|} \hline L^1 & 0 \\ \hline \ell^{2'} & \ell^{3'} \\ \hline L^4 & L^5 \\ \hline \end{array}$$

Since the  $L^1$ ,  $L^4$ ,  $U^1$ , and  $U^2$  blocks of  $L_k$  and  $U_k$  are unaltered through the remainder of the factorization (as well as the first  $k-1$  columns of  $P_k$ ) this information can be read off of the final factors  $L$ ,  $U$ , and  $P$ . After solving for the quantities  $\ell^{2'}$  and  $\ell^{3'}$  the factorization of  $B'$  can begin at the  $k^{th}$

step. We realize a saving of about  $nk^2 - \frac{2}{3}k^3$  multiplications over beginning the factorization completely over. The average number of operations performed is then approximately  $n^3/6$  (if we assume  $k$  has equal likelihood of being one through  $n$ ). Thus a saving of half the computation may be expected.

This updating procedure is the column-wise decomposition analogy of the method given in [2] and [3]. As we see the number of operations required to update is on the order of  $n^3$ . Another procedure, analagous to that given in [1] and [5], provides an updated decomposition with an operation count proportional to  $n^2$ .

For this procedure a different factorization is employed. This has the form

$$L = BU$$

where  $L$  is lower triangular and  $U$  is non-singular (but not necessarily upper triangular). To see that this form is useful for solving the required system, notice that

$$By = BUU^{-1}y = LU^{-1}y = d$$

Thus by solving  $L\bar{y} = d$  and letting  $y = U\bar{y}$  we may solve  $By = d$ . For the transposed system  $B^T w = f$ , we have  $U^T B^T w = L^T w = U^T f$ , thus by letting  $\bar{w} = U^T f$  and solving  $L^T w = \bar{w}$ , we obtain  $w$ . These solutions require about  $3n^2/2$  multiplications compared with  $n^2$  required with the usual LU factorization. The saving in computation is to be gained in the updating of the factorization.

The factorization of the initial  $B$  of the sequence of matrices may be obtained through a simple variant of the columnwise Gaussian elimination algorithm with partial pivoting, yielding  $L = B P \bar{U}$ , where  $P$  is a permutation matrix and  $\bar{U}$  is upper triangular. The initial factor  $U$  is then  $P\bar{U}$  (and we henceforth disregard the fact that  $U$  is permuted upper triangular). This decomposition requires the expected  $n^3/3$  operations of Gaussian elimination.



The updating is accomplished by deleting the  $k^{\text{th}}$  row of  $B$  and inserting the new row  $c$  at the bottom and moving rows  $k+1, \dots, n$  up to  $k, \dots, n-1$ . This (in fact, permuted) matrix is  $B'$ . By deleting row  $k$  of  $L$ , moving rows  $k+1, \dots, n$  up to  $k, \dots, n-1$ , and adding  $cU$  as the final row we have  $\bar{L} = B'U$  where  $\bar{L}$  is lower Hessenberg (and lower triangular in its first  $(k-1) \times (k-1)$  minor). To obtain our final factorization, we need only reduce the super diagonal elements of  $\bar{L}$  to zero using the standard interchange and elimination operations of the Gaussian process. These same operations are performed simultaneously to  $U$ , yielding the factorization  $L' = B'U'$ , where  $L'$  is lower triangular.

The number of operations required for this process is approximately  $n^2 + (n-k)n + (n-k)^2/2 = 5n^2/2 - 2nk + k^2/2$ . Again if we assume that  $k$  is equally likely among the values  $1, \dots, n$  then the number of expected operations is about  $5n^2/3$ . If we include computation for the two solutions,  $By = d$  and  $B^T w = f$ , the total is  $14n^2/3$  compared with  $n^3/6$ , or a reduction by a factor of  $28/n$ . These figures are only for large  $n$  and it is incorrect, as a closer analysis shows, to surmise that small values of  $n$  favor the first algorithm. (Recall that the two  $n^2$  terms for the solution of the systems were ignored in the operation count for the first method.)

Another important consideration in the numerical implementation of the descent algorithm is the ability to iteratively refine solutions. Either of the decompositions presented can be used in the standard fashion to refine solutions of  $By = d$  or  $Bw = f$ , given the residuals computed in extra precision.

It may also be desirable to refine the final solution. Unfortunately, this solution does not occur naturally as the solution to one of these  $n \times n$  systems. It does occur (as does any vertex of the polytope) as part of the solution to an  $(n+1) \times (n+1)$  order system.

At a vertex the set  $M(x)$  of indices of maximum magnitude residual elements has at least  $n+1$  components. If it has more than  $n+1$  (and, has been previously commented, this is extremely rare in practice) there is a subset of exactly  $n+1$  which "define" the vertex: i.e., the  $n+1$  corresponding hyperplanes have only one point, the vertex, in common. We shall henceforth assume  $M(x)$  has exactly  $n+1$  indices.

It must hold that the value  $\theta = |r_i(x)|$  is equal for all  $i \in M(x)$ . We can then say (where  $\bar{b}_i = \text{sgn}(r_i(x))b_i$ )

$$\text{sgn}(r_i(x))a_i x - \theta = \bar{b}_i$$

for  $i \in M(x)$ . Although  $x$  actually enters in a non-linear fashion through  $\text{sgn}(r_i(x))$  we may assume that if the initial value of  $x$  is sufficiently accurate  $r_i(x)$  does not change sign during the refinement.

Select some index  $k \in M(x)$  and solve the transposed linear system to represent  $\text{sgn}(r_k(x))a_k$  as a linear combination of  $\text{sgn}(r_i(x))a_i$ ,  $i \in M(x) \sim k$ . i.e.,

$$\text{sgn}(r_k(x))a_k = \sum_{i \in M(x) \sim k} y_i \text{sgn}(r_i(x))a_i,$$

We then have

$$\begin{aligned} \theta + \bar{b}_k &= \text{sgn}(r_k(x))a_k x = \sum_{i \in M(x) \sim k} y_i \text{sgn}(r_i(x))a_i x \\ &= \sum_{i \in M(x) \sim k} y_i (\theta + \bar{b}_i) = \theta \sum_{i \in M(x) \sim k} y_i + \sum_{i \in M(x) \sim k} y_i \bar{b}_i \\ &= \theta \cdot S + T \end{aligned}$$

where  $S = \sum_{i \in M(x) \sim k} y_i$  and  $T = \sum_{i \in M(x) \sim k} y_i \bar{b}_i$ . As a result  $\theta = (\bar{b}_k - T)/(S - 1)$ . (Notice  $S > 1$  as was shown in the previous section.) To refine the solution, let

$$\Delta \bar{b}_i = \bar{b}_i - \text{sgn}(r_i(x))a_i x + \theta, \quad i \in M(x),$$

for a given  $x$  and  $\theta$ . Then let

$$\Delta\theta = (\Delta\bar{b}_k - \Delta T)/(S-1)$$

(where  $\Delta T = \sum_{i \in M(x) \sim k} y_i \Delta\bar{b}_i$ ) and solve the  $n \times n$  linear system

$$\text{sgn}(r_i(x))a_i \Delta x = \Delta\bar{b}_i + \Delta\theta.$$

Finally  $x$  and  $\theta$  are replaced by  $x - \Delta x$  and  $\theta - \Delta x$ , respectively, and the iteration proceeds until the changes in  $x$  and  $\theta$  are sufficiently small.

The calculation of  $\Delta\bar{b}_i$  and  $\Delta\theta$  are necessarily done in extra precision.

## 5. RESULTS OF NUMERICAL EXPERIENCE

The descent method presented here was programmed in FORTRAN and subjected to a series of tests. The principle subroutine determined the descent step parameter  $\lambda$  and monitored the elements of maximum residual magnitude. Additional subroutine modules implemented the notions of Section 4 for the initial decomposition, updating, and solving of regular and transposed systems.

For comparison purposes the same tests were performed using the ascent algorithm of Bartels and Golub [4] implemented in FORTRAN by Shryer [7]. The form of the test matrices is due to Bartels and Golub [2] and has the elements of the matrix  $A$  as well as the right hand side  $b$  selected randomly as the product  $\eta \cdot v$  where  $\eta$  is uniformly distributed on  $[0,1]$  and  $v$  has the value  $\pm 8^{-1}$ ,  $\pm 8^{-2}$ ,  $\pm 8^{-3}$ ,  $\pm 8^{-4}$  with equal probability. The values of  $n$  were 4, 9, 14, ..., 39, and  $m$  10, 20, ..., 70 with the restriction that  $m > n$ . The descent algorithm (although with a capability for any initial estimate of  $x$ ) used  $x = 0$  as the initial estimate. The Bartels-Golub algorithm determines its own initial estimate.

Storage required for the descent algorithm is about  $3n^2/2 + 3n + 2m$  in addition to that required for  $A$ ,  $x$ , and  $b$ . In the Schryer implementation of the Bartels and Golub algorithm  $mn + n^2 + 6n + m$  locations are required.

Twenty tests were run for each pair of values of  $m$  and  $n$ . Each problem was solved correctly and with comparable accuracy by the two methods. Timings are summarized in the table and graphs (values are average CPU times on the CDC 7600 at the National Center for Atmospheric Research).

In Figure 1, the timings for the descent algorithm are displayed first, the ascent algorithm second. Notice that with the exception of the 20x19 and 30x19 entries, all timings are higher for the ascent method. The ascent

method immediately determines the solution to  $n+1 \times n$  order problems, the descent algorithm does not.

Figure 2 shows the dependency of the timings on  $m$  for  $n = 4$ . Both the algorithms display the expected linear behavior indicating that whether the steps are of the ascent or descent type, the number of steps is proportional to  $m$ . The rate of growth of the ascent timings appears to be about 1.4 of the descent timings.

Figure 3 shows the dependence on  $m$  for  $m = 60$ . The values have been scaled by  $1/n^2$  to feature the growth. The ascent algorithm performs  $O(n^3)$  computations at each of  $O(m)$  steps for a total of  $O(mn^3)$ . Figure 3 suggests this. Alternatively the descent algorithm performs an initial decomposition requiring  $O(n^3)$  operations, then subsequent steps ( $O(m)$  in number) require  $O(n^2)$  operations. Thus, we expect a total of  $O(n^3) + O(mn^2)$  which Figure 3 seems to display.

A version of the ascent algorithm based on the Stiefel method but employing the  $O(n^2)$  updating technique of Section 4 was also programmed and its performance was in general superior to the Bartels-Golub algorithm but inferior to the descent method.

Figure 1

$n \backslash m$	10	20	30	40	50	60
4	2.0 3.6	3.2 5.3	4.5 7.2	5.2 8.9	7.0 10.5	7.6 11.6
9	6.7 8.9	12.7 19.7	16.0 28.1	20.2 33.2	24.0 43.1	26.4 48.6
14		29.1 40.9	38.8 72.1	47.5 92.5	55.8 109.6	61.9 122.3
19		41.6 38.7	72.1 129.6	88.4 177.9	101.5 224.7	116.8 263.3
24			118.1 171.3	141.7 302.1	167.0 346.0	186.0 412.6
29			126.2 100.0	205.0 400.6	256.5 586.5	289.3 735.7
34				287.1 499.8	336.6 835.5	431.0 1134.9

A comparison of average timings for problems with various values of  $m$  and  $n$ . (First figure in box is the descent algorithm timing, second is Bartels-Golub-Schryer algorithm).

Figure 2

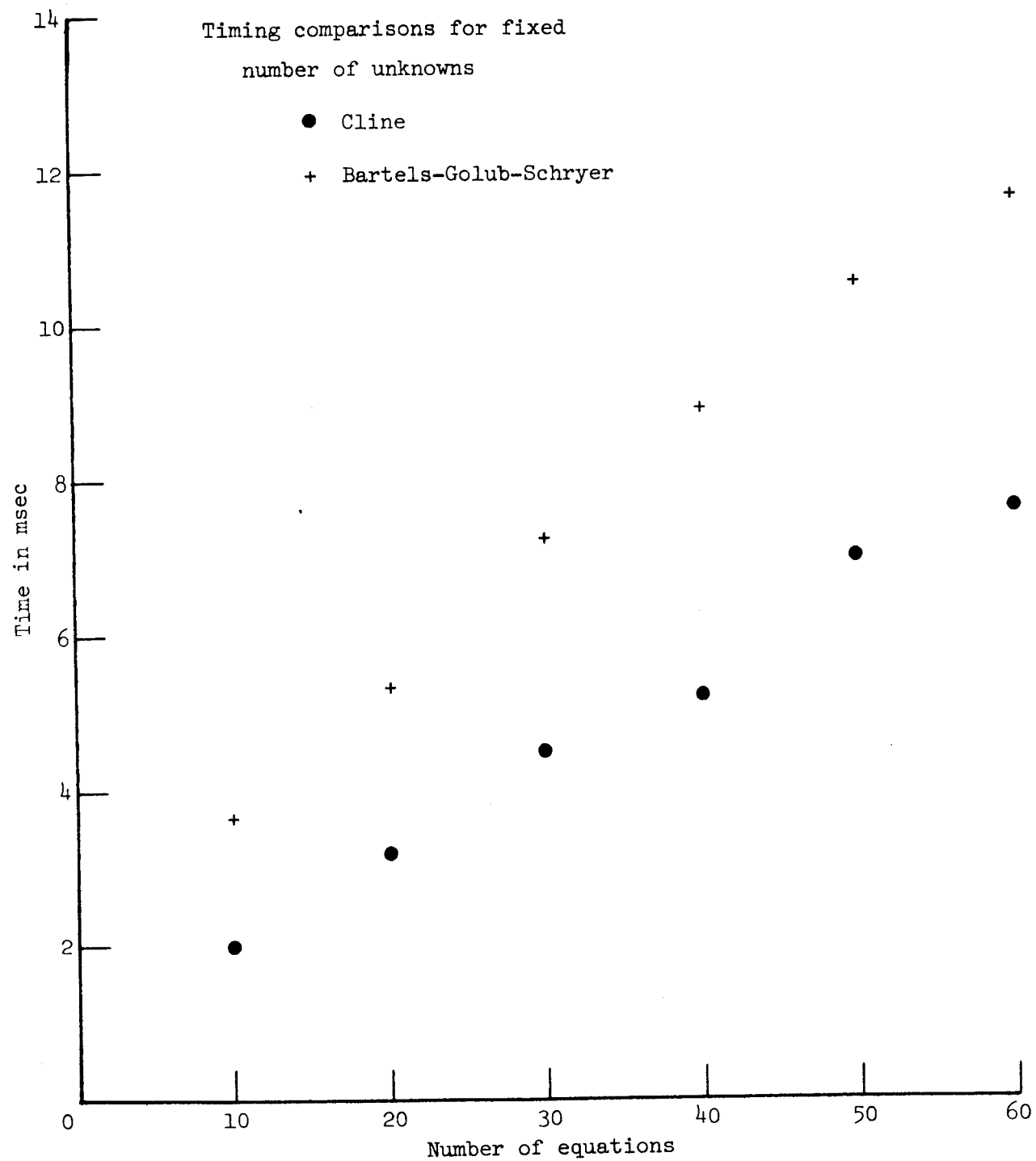
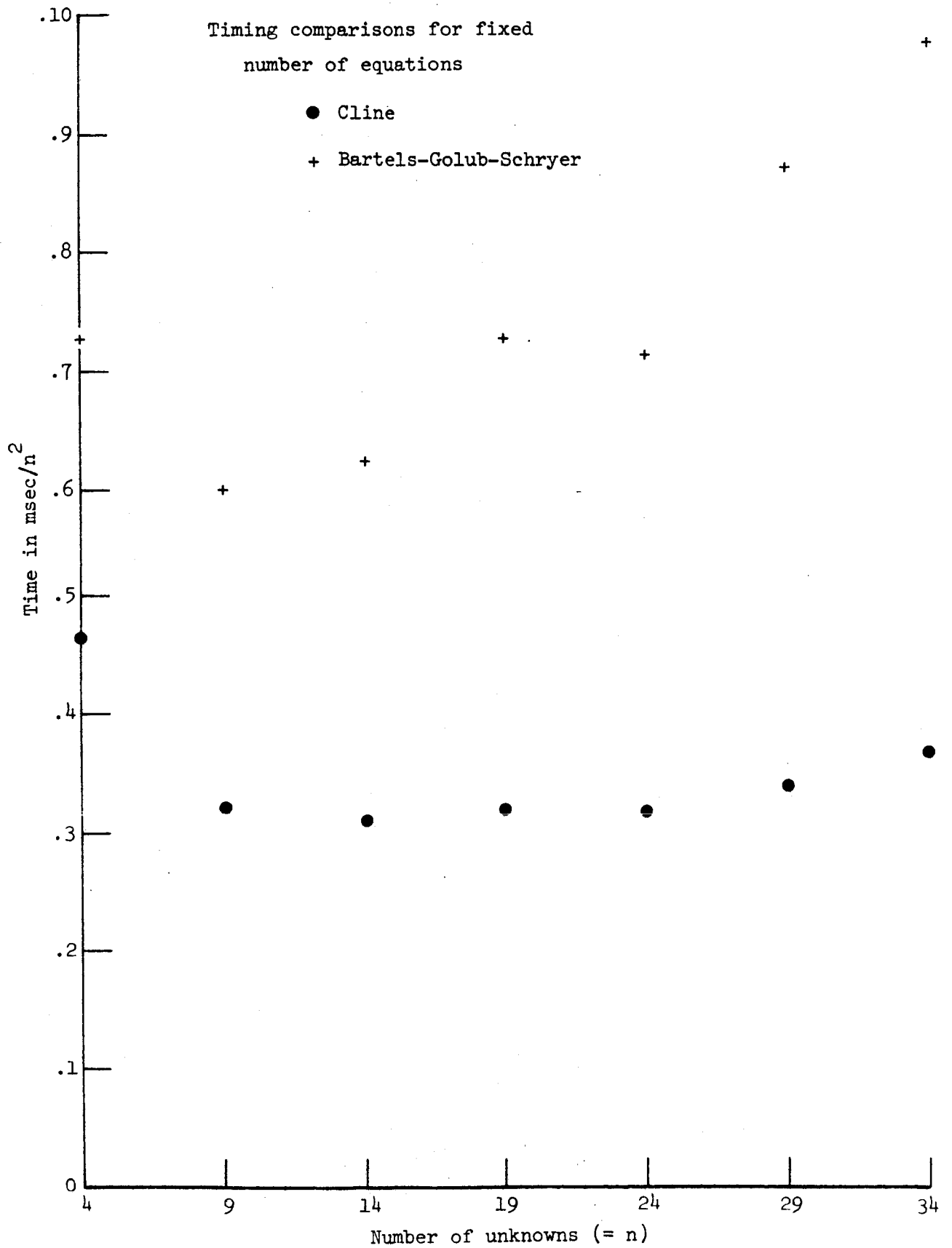


Figure 3





## 6. ADDITION OF LINEAR EQUALITY AND INEQUALITY CONSTRAINTS

The descent method (or any method) easily admits the addition of linear equality constraints. If the problem were to minimize  $||Ax-b||$  such that  $Ex = d$  where  $E$  is  $k \times n$  ( $k \leq n$ ) and  $d$  is a  $k$  vector, then a transformation of coordinates maps this into a standard problem with  $n-k$  variables. To see this, factor  $E$  into the product of  $LU$  where  $L$  is  $k \times n$  lower triangular and  $U$  is  $n \times n$ . Let us assume the equality constraints are independent, in which case  $U$  is non-singular. (Obvious choices for  $U$  are orthogonal or permuted upper triangular matrices.) Letting  $y = Ux$  we see  $Ly = d$  determines the first  $k$  components of  $y$ . Call these components  $y_1$  and the unknown remaining  $n-k$ -vector  $y_2$ . Letting  $A' = AU^{-1}$  (i.e., solving the systems  $a_i'U = a_i$  for the rows of  $A'$ ) and blocking  $A'$  into  $[A_1' A_2']$  with  $k$  and  $n-k$  columns, respectively, we have now  $Ax = A'y = A_1'y_1 + A_2'y_2$ . But  $A_1'y_1$  is known and we are left with determining  $y_2$  which minimizes  $||A_2'y_2 - (b - A_1'y_1)||$ , then finally letting  $x = U^{-1} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ .

For the addition of inequality constraints the problem has the form

$$\text{Minimize } ||Ax - b||$$

$$\text{So that } Gx \geq h.$$

Geometrically, the polytope defined by the problem without the inequality constraints has now been truncated by vertical cuts. Given an "initial feasible vector" (i.e., an initial vector satisfying  $Gx \geq h$ ) the descent occurs as before until the descent step would leave the constrained region. At that point (where  $Gx$  and  $h$  agree in one or more components) constraints enter into the determination of the direction of descent. (Essentially we descend maintaining those component agreements through the descent.) As before a vertex is eventually reached and a modification of the method of Section 3 yields a vertex to vertex descent.

## REFERENCES

1. Bartels, R. H.; J. Stoer; and Ch. Zenger: A Realization of the Simplex Method Based on Triangular Decomposition, Handbook for Automatic Computation, Vol. II, Linear Algebra, J. H. Wilkinson and C. Reinsch, Eds., Springer-Verlag, New York, 1971, pp. 152-190.
2. Bartels, R. H. and G. H. Golub: Computational Considerations Regarding the Calculation of Chebyshev solutions for Over-determined Linear Equations Systems by the Exchange Method, Tech. Rep. No. CS67, Comput. Sch. Dep., Stanford U., Stanford, Calif.
3. Bartels, R. H. and G. H. Golub: Stable Numerical Methods for Obtaining the Chebyshev Solution to an Overdetermined System of Equations, CACM 11 (June 1968), pp. 403-408.
4. Bartels, R. H. and G. H. Golub: Algorithm 328: Chebyshev Solution to an Overdetermined Linear System, CACM 11 (June 1968), pp. 428-430.
5. Bartels, R. H.; G. H. Golub, and M. A. Saunders: Numerical Techniques in Mathematical Programming, Tech. Rep. No. CS162, Comput. Sci. Dep. Stanford, Stanford U., Calif.
6. Berge, C.: Topological Spaces, MacMillan, New York, 1963.
7. Cheney, E. W.: Introduction to Approximation Theory, McGraw-Hill, New York, 1966.
8. Schryer, N. L.: Certifications of Algorithm 328: Chebyshev Solution to an Overdetermined Linear System, CACM 12 (June 1969), p. 326.
9. Stiefel, E. L.: Über Diskrete und Lineare Tschebyscheff - Approximationen, Numer. Math. 1 (1959), pp. 1-28.